Exercice 1. Moyenne arithmético-géométrique

Soient a et b deux réels tels que $0 \leqslant a \leqslant b$. On définit les suites $(u_n)_n$ et $(v_n)_n$ par récurrence de la manière suivante :

$$\begin{cases} u_0 = a \\ v_0 = b \end{cases} \quad \text{et pour tout } n \geqslant 0 : \quad \begin{cases} u_{n+1} = \sqrt{u_n v_n} \\ v_{n+1} = \dfrac{u_n + v_n}{2} \end{cases}$$

On admet que la suite $(u_n)_n$ est croissante, que la suite $(v_n)_n$ est décroissante et que ces deux suites ont la même limite. La limite commune de $(u_n)_{n\in\mathbb{N}}$ et $(v_n)_{n\in\mathbb{N}}$ est appelée la moyenne arithmético-géométrique de a et b et sera notée $M(a,b)$ dans la suite. Par exemple :

a, b	$1,1$	$2,8$	$3,10$
$M(a,b)$	1	$4.486057160575\ldots$	$5.977670553300\ldots$

Écrire une fonction `approx_moy_ag` qui prend en argument deux flottants a, b ainsi qu'un entier n, et renvoie le couple (u_n, v_n). Vérifiez que lorsque n est grand, votre fonction renvoie des valeurs cohérentes avec les valeurs données dans le tableau ci-dessus. Vérifiez également que :

`n`	0	1	2
`approx_moy_ag(5,20,n)`	`(5, 20)`	`(10.0, 12.5)`	`(11.1803..., 11.25)`

Exercice 2. Parcours de listes

1. Écrire une fonction qui renvoie le maximum d'une liste d'entiers. Votre fonction déclenchera une erreur si la liste est vide.

`L`	`[]`	`[1,2,3]`	`[-8,6,0,4]`	`[8,6,4,0]`	`[-4,-3,-8,-1]`
`maximum(L)`	Erreur	3	6	8	-1

2. Écrire une fonction `est_croissante` qui prend en entrée une liste d'entiers `L` et renvoie `True` si `L` est croissante et `False` sinon. Vérifiez que votre fonction renvoie `True` pour les listes :

   `[]` `[0]` `[1,2,3]` `[1,1,1]` `[-10,-3,0,0,4,8,8]`

   mais renvoie `False` pour :

   `[0,-1]` `[-3,4,5,4,5,6]` `[1,0,1,2,3,4]` `[-2,-1,0,1,2,3,2]`

3. À l'aide d'une boucle `for`, écrire une fonction `inverser` qui prend en entrée `L` une liste d'entiers et renvoie une liste contenant les mêmes éléments dans l'ordre inverse. Par exemple :

`L`	`[]`	`[0]`	`[3,2,1]`	`[5,8,4,1,0,2,1]`
`inverser(L)`	`[]`	`[0]`	`[1,2,3]`	`[1,2,0,1,4,8,5]`

Exercice 3. Compréhensions de listes

Dans cet exercice, il est demandé d'utiliser des compréhensions de listes.

1. À l'aide d'une compréhension de liste, écrire une fonction `mult_elem_liste` qui prend en paramètre un nombre a et une liste d'entiers `L`, et renvoie la liste où chaque élément de `L` a été multiplié par a. Par exemple :

a	L	mult_elem_liste(a, L)
2	[1,2,3]	[2, 4, 6]
0.5	[]	[]
0.5	[10, 8, 6, -8, 2, 7]	[5.0, 4.0, 3.0, -4.0, 1.0, 3.5]
-4	[7, 4, 1, 0, -7, -4, -1]	[-28, -16, -4, 0, 28, 16, 4]

2. À l'aide d'une compréhension de liste, écrire une fonction `val_abs` qui prend en entrée une liste de nombres flottants `L` et renvoie la liste obtenue en prenant la valeur absolue de chaque élément de `L`. Par exemple :

L	[]	[1, 2, 3]	[-5, 3, 0, 5]	[-4]
val_abs(L)	[]	[1, 2, 3]	[5, 3, 0, 5]	[4]

3. À l'aide d'une compréhension de liste, écrire une fonction `prod_couples` qui prend en entrée une liste de couples d'entiers et renvoie une liste d'entiers dont chaque élément est le produit des éléments du couple de même indice. Par exemple :

L	[]	[(8,8)]	[(1,2), (-5,2), (0,0), (9,3)]
prod_couples(L)	[]	[64]	[2, -10, 0, 27]

Exercice 4. Crible d'Ératosthène

Le crible d'Ératosthène est un algorithme qui calcule tous les nombres premiers compris entre 0 et un paramètre `n` $\in \mathbb{N}$. Pour cela, on utilise une liste de booléens `L` de taille `n+1` telle que `L[i]` vaut `False` lorsqu'on a trouvé un nombre premier qui divise `i` et `True` sinon. Voici les étapes du crible d'Ératosthène :

- Initialement, toutes les cases de `L` valent `True` sauf `L[0]` et `L[1]` qui valent `False`.
- Lors de la première étape, on déclare que le nombre `2` est premier car c'est le premier indice de `L` dont la case vaut `True`. On met ensuite à jour `L` en affectant `False` aux cases dont l'indice est un multiple de `2`. On affecte donc `False` aux cases d'indices `2`, `4`, `6`, `8`, ...
- Lors de la deuxième étape, on déclare que le nombre `3` est premier car c'est le premier indice de `L` dont la case vaut `True`. On met ensuite à jour `L` en affectant `False` aux cases dont l'indice est un multiple de `3`. On affecte donc `False` aux cases d'indices `3`, `6`, `9`, `12`, ...
- Lors de la troisième étape, on déclare que le nombre `5` est premier car c'est le premier indice de `L` dont la case vaut `True`. On met ensuite à jour `L` en affectant `False` aux cases dont l'indice est un multiple de `5`. On affecte donc `False` aux cases d'indices `5`, `10`, `15`, `20`, ...
- Plus généralement à chaque nouvelle étape, on identifie le plus petit indice `i` tel que `L[i]` vaut `True`, on déclare que le nombre `i` est premier et on affecte à `False` toutes les cases de `L` dont l'indice est un multiple de `i`.

Écrire une fonction `crible_eratosthene` qui prend en entrée un entier `n` et qui renvoie une liste `R` contenant tous les nombres premiers compris entre `0` et `n`. Vérifiez que :

n	0	1	2	12	13
crible_eratosthene(n)	[]	[]	[2]	[2,3,5,7,11]	[2,3,5,7,11,13]

Exercice 5. The Minion Game

Ce problème est issu du site HackerRank :

`https://www.hackerrank.com/challenges/the-minion-game/problem`.

Kevin et Stuart veulent jouer à un jeu.

Règles du jeu. On donne à chaque joueur la même chaîne de caractères `s`. Les deux joueurs doivent créer des sous-chaînes de caractères en utilisant les lettres de `s`. Stuart doit faire des mots commençant par des consonnes. Kevin doit faire des mots commençant par des voyelles. Le jeu s'arrête lorsque les deux joueurs ont fait toutes les sous-chaînes possibles.

Décompte des points. Chaque joueur obtient +1 point pour chaque occurrence de sa sous-chaîne dans la chaîne `s`.

Par exemple. Avec la chaîne de caractères `s = "BANANA"`. Mot de Kevin commençant par une voyelle = `ANA`. Ici, `ANA` apparaît deux fois dans `BANANA`. Donc Kevin obtient 2 points. Pour une meilleur compréhension, voir l'image ci-dessous :

Votre but est de déterminer le gagnant du jeu et son score.

Description de la fonction. La fonction s'appellera `minion_game`. `minion_game` prend en argument :

- `s` de type `str` : la chaîne de caractères à analyser.

Affichage. Votre fonction affichera le nom du gagnant et son score, séparés par un espace sur une seule ligne, ou `Draw` s'il n'y a pas de gagnant.

Remarque. La chaîne de caractères `s` ne contient que des lettres majuscules : de `A` à `Z`.

Exemple. Sur l'entrée `"BANANA"`, votre fonction affichera `Stuart 12`.

Exercices à rendre au plus tard le 01/10/2023 à 20h

Les consignes pour les rendus sont données dans le TP 1. Si vous ne les respectez pas, vous perdrez des points.

Exercice 6. Moyenne et écart type

Soit $E = \{x_1, \dots, x_n\}$ un ensemble de nombres réels. La moyenne et l'écart type des éléments de E, notés respectivement $\overline{x}$ et σ, sont définis par :

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad\qquad \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

1. Écrire une fonction `moyenne` qui renvoie la moyenne des éléments d'une liste de flottants. Votre fonction déclenchera une erreur si la liste est vide. Voir les exemples ci-dessous.
2. Écrire une fonction `ecart_type` qui renvoie l'écart type des éléments d'une liste de flottants. Votre fonction déclenchera une erreur si la liste est vide. Voir les exemples ci-dessous.

`L`	`[]`	`[1, 2, 3]`	`[1, 8, 10, 4, 80]`	`[-5, 6]`
`moyenne(L)`	Erreur	`2.0`	`20.6`	`0.5`
`ecart_type(L)`	Erreur	`0.816 ...`	`29.863 ...`	`5.5`

Exercice 7. Salaire d'un employé

Le salaire de base d'un employé est de 10€03 de l'heure. Lorsque l'employé effectue plus de 35 heures par semaine, le salaire horaire est majoré en suivant les règles suivantes :

- Les 35 premières heures sont toujours rémunérées 10€03 chacune (même si l'employé a effectué plus de 35 heures).
- Les 8 premières heures supplémentaires (de la $36^{\text{ème}}$ à la $43^{\text{ème}}$ heure travaillée) sont rémunérées 25% de plus que le salaire horaire de base.
- Les autres heures supplémentaires sont rémunérées 50% de plus que le salaire horaire de base.

Dans le cas où le salaire ne tombe pas juste, il est arrondi au centime d'euro supérieur. Pour cela, on pourra utiliser la fonction `ceil` du module `math` :

```
import math
### Cette fonction arrondit un prix au centime d'euros superieur
def arrondi(s):
    return (math.ceil(s*100))/100
```

Écrire une fonction `salaire` qui prend en entrée un entier représentant le nombre d'heures travaillées dans la semaine, et renvoie la rémunération de l'employé. Par exemple :

`s`	0	5	35	36	40	43	44	80
`salaire(s)`	0.0	50.15	351.05	363.59	413.74	451.35	466.4	1008.02

Exercice 8. Nombres parfaits et nombres amis

Pour un entier naturel $n \in \mathbb{N}^*$, on appelle diviseur propre un diviseur de n différent de n.

1. Écrire une fonction `div_propres` qui prend en entrée un entier `n` et renvoie la liste de ses diviseurs propres.

Un entier naturel $n \in \mathbb{N}^*$ est dit ***parfait*** s'il est égal à la somme de ses diviseurs propres.

2. Écrire une fonction `est_parfait` qui prend entrée un entier `n`, qui renvoie `True` si `n` est parfait et `False` s'il ne l'est pas. Par exemple vérifiez que 6, 28 et 496 sont parfaits alors que 1, 2, 740 et 12345 ne le sont pas.

Deux entiers $n_1 \in \mathbb{N}^*$ et $n_2 \in \mathbb{N}^*$ sont dits ***amis*** si la somme des diviseurs propres de n_1 est égale à n_2 et vice-versa.

3. Écrire une fonction `afficher_amis` (fonction sans argument) qui affiche tous les couples (n_1, n_2) tels que $n_1, n_2 \in [\![1, 20\ 000]\!]$ sont deux nombres amis avec $n_1 \leqslant n_2$. Vérifier que votre fonction affiche 12 couples et qu'elle n'affiche pas `None`. Si votre fonction est trop lente pour afficher les 12 couples, c'est que votre stratégie de résolution n'est pas suffisamment efficace.

Exercice 9. Course de grenouilles (exercice facultatif)

Cet exercice est issu du site France-ioi :

`https://www.france-ioi.org/algo/task.php?idChapter=656&idTask=2279`

Aujourd'hui est organisée la grande course de grenouilles. Chaque candidat a entraîné sa grenouille durement toute l'année pour ce grand événement. La course se déroule en tours et, à chaque tour, une question est posée aux dresseurs. Le premier qui trouve la réponse gagne le droit d'ordonner à sa grenouille de faire un bond. Dans les règles de la course, il est stipulé que c'est la grenouille qui restera le plus longtemps en tête qui remportera la victoire.

Ce que doit faire votre programme. n grenouilles numérotées de 1 à n sont placées sur une ligne de départ. À chaque tour, on vous indique le numéro de la seule grenouille qui va sauter lors de ce tour, et la distance qu'elle va parcourir en direction de la ligne d'arrivée.

Écrivez un programme qui détermine laquelle des grenouilles a été strictement en tête de la course au début du plus grand nombre de tours. Notez que comme on s'intéresse à qui est en tête au début de chaque tour, le bond du dernier tour ne sert à rien car même si la grenouille concernée passe en tête, la course est finie (il est purement honorifique selon la tradition de la course).

Entrée. Votre fonction `course` prend en entrée :

- n, le nombre de grenouilles qui participent à la course. Les grenouilles sont numérotées de 1 à n.
- Une liste de couples « `L: list[int,int]` ». Chaque élément de `L` décrit un tour par deux entiers. Le premier entier est le numéro de la grenouille qui saute à ce tour, et le deuxième, est la distance parcourue par la grenouille lors de ce saut.

Sortie. Vous devez renvoyer un entier : le numéro de la grenouille qui a été strictement en tête au début du plus grand nombre de tours. En cas d'égalité entre plusieurs grenouilles, choisissez celle dont le numéro est le plus petit.

Exemple. Avec $n = 4$ et `L = [(2,2), (1,2), (3,3), (4,1), (2,2), (3,1)]`, votre fonction doit renvoyer 2.

Commentaires. Pour l'exemple proposé, le tableau ci-contre indique la distance totale parcourue par chaque grenouille au début de chaque tour. La grenouille 1 est restée 0 tour strictement en tête, la seconde 2 (les tours 2 et 6), la troisième 2 (les tours 4 et 5) et la quatrième 0. C'est donc la grenouille 2 qui remporte la victoire.

Grenouille	1	2	3	4
Tour 1	0	0	0	0
Tour 2	0	2	0	0
Tour 3	2	2	0	0
Tour 4	2	2	3	0
Tour 5	2	2	3	1
Tour 6	2	4	3	1

Exercices à rendre au plus tard le 08/10/2023 à 20h

Exercice 10. Compréhensions de listes

Dans cet exercice, il est demandé d'utiliser des compréhensions de listes.

1. À l'aide d'une compréhension de liste, écrire une fonction `est_pair` qui prend en entrée une liste d'entiers `L1` et renvoie une liste de booléens `L2` telle que pour `i` $\in \{0, \ldots, \texttt{len(L1)-1}\}$:

$$\begin{cases} \texttt{L2[i]} \text{ vaut } \texttt{True} \text{ lorsque } \texttt{L1[i]} \text{ est pair.} \\ \texttt{L2[i]} \text{ vaut } \texttt{False} \text{ lorsque } \texttt{L1[i]} \text{ est impair.} \end{cases}$$

   Vérifiez que :

   `est_pair([])` vaut `[]`,
   `est_pair([1, 2, 3, 4, 5, 6])` vaut `[False, True, False, True, False, True]`,
   `est_pair([-5, 6, 7, 0])` vaut `[False, True, False, True]`.

2. À l'aide d'une compréhension de liste, écrire une fonction `liste_diviseurs` qui prend en entrée un entier `n` $\geqslant 0$, et renvoie la liste des entiers de $[\![1, \texttt{n}]\!]$ qui divisent `n`. Par exemple :

`n`	`0`	`11`	`24`	`95`
`liste_diviseurs(n)`	`[]`	`[1, 11]`	`[1, 2, 3, 4, 6, 8, 12, 24]`	`[1, 5, 19, 95]`

Exercice 11. Mélange de deux listes

Écrire une fonction `melange` qui prend en entrée deux listes L_1 et L_2 et renvoie la liste :

$$L = [L_1[0], L_2[0], L_1[1], L_2[1], L_1[2], L_2[2], \ldots].$$

Dans le cas où L_1 et L_2 ne sont pas de la même longueur, votre programme ajoutera les éléments en surplus à la fin de la liste `L`. Par exemple :

`melange([], [])` vaut `[]`,
`melange([], [1,2,3])` vaut `[1, 2, 3]`,
`melange([1,2,3], [])` vaut `[1, 2, 3]`,
`melange([1,2,3], [4,5,6])` vaut `[1, 4, 2, 5, 3, 6]`,
`melange([1,2,3], [4,5,6,7,8,9])` vaut `[1, 4, 2, 5, 3, 6, 7, 8, 9]`,
`melange([1,2,3,4,5,6], [7,8,9])` vaut `[1, 7, 2, 8, 3, 9, 4, 5, 6]`.

Exercice 12. Listes d'entiers de même signe

Écrire une fonction `ont_meme_signe` qui prend en entrée une liste d'entiers et renvoie `True` si tous les éléments sont positifs ou bien s'ils sont tous négatifs. Attention, 0 est considéré comme positif et négatif. Par exemple, votre fonction doit renvoyer `True` pour :

`[]`, `[0, 1, 2, 3]`, `[-1, 0, -2]`, `[0, 0]`.

et `False` pour :

`[-1, -2, -3, 0, 5]`, `[0, 0, 1, -1]`.

Exercice 13. Composition musicale (exercice facultatif)

Cet exercice est issu du site France-ioi :

`http://www.france-ioi.org/algo/task.php?idChapter=656&idTask=2240`

Écouter de la musique peut être très agréable mais lorsqu'un morceau est vraiment très répétitif, il arrive parfois qu'on s'ennuie un peu. Aussi le professeur de composition musicale du conservatoire a décidé d'imposer une règle très stricte : quand il relit les morceaux composés par ses élèves, dès qu'il voit deux notes identiques côte à côte, il les efface toutes les deux ! Il continue ainsi d'effacer tant qu'il existe deux notes égales consécutives. Ce travail étant long et fastidieux, il se demande s'il n'est pas possible de l'automatiser.

Ce que doit faire votre fonction : Les notes de musiques sont représentées par les lettres `a`, `b`, `c`, `d`, `e`, `f` et `g`. Votre fonction appelée `elimination_doublons` doit prendre en entrée une chaîne de caractères représentant le morceau de musique et doit renvoyer la version du morceau "corrigée" où tous les doublons sont supprimés tant qu'il en existe.

Exemple Sur l'entrée `"baaabbacddc"`, votre fonction doit renvoyer `"b"`. Une suite possible d'élimination des doublons est la suivante :

`"baaabbacddc"` $\rightsquigarrow$ `"baaabbacc"` $\rightsquigarrow$ `"babbacc"`
$\rightsquigarrow$ `"babba"` $\rightsquigarrow$ `"baa"` $\rightsquigarrow$ `"b"`